CLAIMS

I claim:

1. A method of removing empty string terms from an automaton A having a set of states "p", a set of states "q", and a set of outgoing transitions from the set of states "p", E[p], the method comprising:

computing an ε-closure for each state "p" of the automaton A;

modifying E[p] by:

removing each transition labeled with an empty string; and

adding to E[p] a non-empty-string transition, wherein each state "q" is left with its weights pre-multiplied by an ε-distance from state "p" to a state "q" in the automaton A.

2. The method of claim 1, further comprising:

removing inaccessible states using a depth-first search of the automaton A.

3. The method of claim 1, wherein adding to E[p] non-empty-string transitions further comprises leaving q with weights (d[p,q] $\otimes$ ρ[q]) to E[p].

4. The method of claim 1, wherein the step of computing ε-closure for each input state of an input automaton A further comprises:

removing all transitions not labeled with an empty string from automaton A to produce an automaton A_ε;

decomposing A_ε into its strongly connected components; and

computing all-pairs shortest distances in each component visited in reverse topological order.

5. The method of claim 1, wherein the step of computing ε-closure for each input state of an input automaton A further comprises:

decomposing A_ε into its strongly connected components;

performing a single-source shortest-distance algorithm according to the following pseudo code:

```
for each p ∈ Q
      do d[p] ← r[p] ← 0̄
d[s] ← r[s] ← 1̄
S ← {s}
while S ≠ 0
      do q ← head [S]
         DEQUEUE (S)
         r ← r[q]
         r[q] ← 0̄
         for each e ∈ E[q]
            do if d[n[e]] ≠ d[n[e]] ⊕ (r ⊗ w[e])
               then d[n[e]] ← d[n[e]] ⊕ (r ⊗ w[e])
                  r[n[e]] ← r[n[e]] ⊕ (r ⊗ w[e])
                  if n[e] ∉ S
                     then ENQUEUE (S, n[e])
d[s] ← 1̄
```

6. The method of claim 1, wherein the step of computing the ε-closure for each state "p" further comprises computing each the ε-closure according to the following equation:

$$C[p] = \{(q,w) : q \in \epsilon[p], d[p,q] = w \in K - \{\bar{0}\}\}.$$

7. The method of claim 6, wherein the step of modifying outgoing transitions of each state "p" further comprises modifying the outgoing transitions of each state p according to the following procedure:

```
for each p ∈ Q
  do E[p] ← {e ∈ E[p] : i[e] ≠ ϵ}
    for each (q,w) ∈ C[p]
      do E[p] ← E[p] ∪ {(p,a,w ⊗ w',r) : (q, a, w', r) ∈ E[q], a ≠ ϵ}
        if q ∈ F
          then if p ∉ F
            then F ← F ∪ {p}
              ρ[p] ← ρ[p] ⊕ (ω ⊕ ρ[q])
```

8. The method of claim 7, wherein a state is a final state if some state "q" within a set of states reachable from "p" via a path labeled with an empty string is final and the final weight is then: $\rho[p] = \bigoplus_{q \in e[p] \cap F} (d[p,q] \otimes \rho[q])$.

9. The method of claim 8, further comprising:

performing a depth-first search of the automaton A after removing the empty strings.

10. A method of producing an equivalent weighted automaton "B" with no ε-transitions for any input weighted automaton "A" having at least one ε-transition, the automaton "A" having a set of states "p", and a set of states "q", the method comprising:

computing an ε-closure for each state "p" of the input weighted automaton "A";

modifying outgoing transitions of each state "p" by:

removing each transition labeled with an empty string; and

adding to each transition leaving state "p" a non-empty-string transition, wherein each state "q" is left with its weights pre-multiplied by an

ε-distance from state "p" to "q" in the automaton "A" to produce the automaton "B" equivalent to automaton A without the ε-transitions.

11. The method of claim 10, further comprising:

removing inaccessible states using a depth-first search of the automaton.

12. The method of claim 11, wherein adding to the outgoing transitions non-empty-string transitions further comprises leaving each state "q" with weights $(d[p,q] \otimes \rho[q])$ to the transitions leaving p.

13. A method of claim 10, wherein the step of computing an ε-closure for each input state of an input automaton "A" further comprises:

removing all non-ε-transitions to produce an automaton A_ε;

decomposing A_ε into its strongly connected components; and

computing all-pairs shortest distances in each component visited in reverse topological order.

14. The method of claim 10, wherein the step of computing the ε-closure for each state "p" further comprises computing each of the ε-closures according to the following equation:

$$C[p] = \{(q,w) : q \in \epsilon[p], d[p,q] = w \in K - \{\bar{0}\}\}.$$

15. The method of claim 14, wherein the step of modifying outgoing transitions of each state "p" further comprises modifying the outgoing transitions of each state p according to the following procedure:

```
(1)   for each p ∈ Q
(2)     do E[p] ← {e ∈ E [p] : i[e] ≠ ϵ}
(3)       for each (q,w) ∈ C[p]
(4)         do E[p] ← E[p] ∪ {(p,a,w ⊗ w',r) : (q, a, w', r) ∈ E[q],a ≠ ϵ}
(5)           if q ∈ F
(6)             then if p ∉ F
(7)               then F ← F ∪ {p}
(8)                 ρ[p] ← ρ[p] ⊕ (ω ⊕ ρ[q]).
```

16. A method of producing an automaton B from an automaton A, the automaton B having no empty string transitions, the method comprising:

computing for each state p in automaton A its ε-closure C[p] according to the following: $C[p] = \{(q,w) : q \in \epsilon[p], d[p,q] = w \in K - \{\bar{0}\}\}$, where $\epsilon[p]$ represents states labeled with an empty string;

removing each transition labeled with an empty string; and

adding to each transition leaving state "p" a non-empty-string transition, wherein each state "q" is left with its weights pre-multiplied by an ε-distance from state "p" to "q" in the automaton "A" to produce the automaton "B" equivalent to automaton A without the ε-transitions.

17. The method of claim 16, wherein adding non-empty strings to E[p] is performed according to the following code:

```
(1)   for each p ∈ Q
(2)     do E[p] ← {e ∈ E [p] : i[e] ≠ ϵ}
(3)       for each (q,w) ∈ C[p]
(4)         do E[p] ← E[p] ∪ {(p,a,w ⊗ w',r) : (q, a, w', r) ∈ E[q],a ≠ ϵ}
(5)           if q ∈ F
(6)             then if p ∉ F
(7)               then F ← F ∪ {p}
(8)                 ρ[p] ← ρ[p] ⊕ (ω ⊕ ρ[q]),
```

18. The method of claim 10, further comprising modifying E[p] according to the following procedure:

$$
\begin{array}{l}
\textbf{for } \text{each } p \in Q \\
\quad \text{do } E[p] \leftarrow \{e \in E[p] \ : \ i[e] \neq \epsilon\} \\
\quad\quad \textbf{for } \text{each } (q,w) \in C[p] \\
\quad\quad\quad \text{do } E[p] \leftarrow E[p] \cup \{(p,a,w \otimes w',r) : (q, a, w', r) \in E[q], a \neq \epsilon\} \\
\quad\quad\quad\quad \textbf{if } q \in F \\
\quad\quad\quad\quad\quad \textbf{then if } p \notin F \\
\quad\quad\quad\quad\quad\quad \textbf{then } F \leftarrow F \cup \{p\} \\
\quad\quad\quad\quad\quad\quad\quad \rho[p] \leftarrow \rho[p] \oplus (\omega \oplus \rho[q])
\end{array}
$$

19. A method of producing an equivalent weighted automaton "B" with no ε-transitions for any input weighted automaton "A" having a set of transitions E, wherein each transition "e" in the set of transitions has an input label i[e], at least one transition being an ε-transition, a set of states P, each state in the set of states P is denoted as "p", and a set of states Q, each state in the set of states Q denoted as "q", a weight w[e] for each transition "e", and E[p] the transitions leaving each state "p" and E[q] being the transitions leaving state "q", an ε-closure for a state being defined as C[p], and where ε[p] represents a set of states reachable from state "p" via a path labeled with an ε-transition, the method comprising:

computing an ε-closure C[p] for each state "p" of the input weighted automaton "A";

removing each ε-transition to produce an automaton A_ε; and

adding to E[p] non-empty-string transitions leaving each state "q" from the set of states reachable from "p" via a path labeled with an ε-transitions wherein each state "q" is left with its weights pre-multiplied by an ε-distance from state "p" to "q" in the

automaton "A" to produce the automaton "B" equivalent to automaton A without ε-transitions.

20. A method of producing an equivalent weighted automaton "B" with no ε-transitions for any input weighted automaton "A" having a set of transitions "e", at least one of which is an ε-transition, a set of states "p", and a set of states "q", the method comprising:

computing an ε-closure C[p] for each state "p" of the input weighted automaton "A";

for each state "p", determining the non-ε-transitions from state "p";

for each state "q" having a weight "w" within the computed ε-closure C[p]:

adding to E[p] the non-ε-transitions leaving each state "q"; and

if state "q" is part of a set of final states F, and if state "p" is not part of the set of final states F:

defining state "p" as included within the set of final states "F" and the final weight $\rho[p]$ as pre-⊗-multiplied by w, the ε-distance from state "p" to state "q" in the automaton A.

21. A method of removing string terms "a" from an automaton A having a set of states "p", a set of states "q", and a set of outgoing transitions from the set of states "p", the method comprising:

computing an a-closure for each state "p" of the automaton A;

modifying E[p] by:

removing each transition labeled with a string term "a"; and

adding to E[p] a non-"a"-string transition, wherein each state "q" is left with its weights pre-⊗-multiplied by an a-distance from state "p" to a state "q" in the automaton A.

22. The method of claim 21, further comprising:

removing inaccessible states using a depth-first search of the automaton A.

23. The method of claim 21, wherein adding to E[p] non-"a"-string transitions further comprises leaving q with weights $(d[p,q] \otimes \rho[q])$ to E[p].

24. The method of claim 21, wherein the step of computing an a-closure for each input state of an input automaton A further comprises:

removing all transitions not labeled with a string "a" from automaton A to produce an automaton A_a;

decomposing A_a into its strongly connected components; and

computing all-pairs shortest distances in each component visited in reverse topological order.

25. The method of claim 21, wherein the step of computing an a-closure for each input state of an input automaton A further comprises:

decomposing A_a into its strongly connected components;

performing a single-source shortest-distance algorithm according to the following pseudo code:

for each $p \in Q$
  do $d[p] \leftarrow r[p] \leftarrow \bar{0}$
$d[s] \leftarrow r[s] \leftarrow \bar{1}$
$S \leftarrow \{s\}$

```
while S ≠0
    do q ← head [S]
       DEQUEUE (S)
       r ← r[q]
       r[q] ← Ō
       for each e ∈ E[q]
          do if d[n[e]] ≠ d[n[e]] ⊕ (r ⊗ w[e])
             then d[n[e]] ← d[n[e]] ⊕ (r ⊗ w[e])
                  r[n[e]] ← r[n[e]] ⊕ (r ⊗ w[e])
                  if n[e] ∉ S
                     then ENQUEUE (S, n[e])
d[s] ← 1̄
```

26. The method of claim 21, wherein the step of computing the a-closure for each state "p" further comprises computing each of the a-closures according to the following equation:

$$C[p] = \{(q,w) : q \in a[p], d[p,q] = w \in K - \{\bar{0}\}\}.$$

27. The method of claim 26, wherein the step of modifying outgoing transitions of each state "p" further comprises modifying the outgoing transitions of each state p according to the following procedure:

```
for each p ∈ Q
  do E[p] ← {e ∈ E [p] : i[e] ≠ a}
    for each (q,w) ∈ C[p]
        do E[p] ← E[p] ∪ {(p,a,w ⊗ w',r) : (q, a, w', r) ∈ E[q],a ≠ a}
          if q ∈ F
             then if p ∉ F
                then F ← F ∪ {p}
                   ρ[p] ← ρ[p] ⊕ (ω ⊕ ρ[q])
```

28. The method of claim 27, wherein a state is a final state if some state "q" within a set of states reachable from "p" via a path labeled with an empty string is final and the final weight is then: $\rho[p] = \bigoplus_{q \in e[p] \cap F} (d[p,q] \otimes \rho[q])$.

29. The method of claim 28, further comprising:

performing a depth-first search of the automaton A after removing the "a" strings.

30. A method of removing empty string terms from a transducer A having a set of states "p", a set of states "q", and a set of outgoing transitions from the set of states "p", E[p], the method comprising:

computing an ε-closure for each state "p" of the transducer A;

modifying E[p] by:

removing each transition labeled with an empty string; and

adding to E[p] a non-empty-string transition, wherein each state "q" is left with its weights pre-multiplied by an ε-distance from state "p" to a state "q" in the transducer A.

31. The method of claim 30, further comprising:

removing inaccessible states using a depth-first search of the transducer A.

32. The method of claim 30, wherein adding to E[p] non-empty-string transitions further comprises leaving q with weights ($d[p,q] \otimes \rho[q]$) to E[p].

33. The method of claim 30, wherein the step of computing ε-closure for each input state of an input transducer A further comprises:

removing all transitions not labeled with an empty string from transducer A to produce a transducer A_ε;

decomposing A_ε into its strongly connected components; and

computing all-pairs shortest distances in each component visited in reverse topological order.

34. The method of claim 30, wherein the step of computing ε-closure for each input state of an input transducer A further comprises:

decomposing A_ε into its strongly connected components;

performing a single-source shortest-distance algorithm according to the following pseudo code:

```
for each p ∈ Q
    do d[p] ← r[p] ← 0̄
d[s] ← r[s] ← 1̄
S ← {s}
while S ≠ 0
    do q ← head [S]
       DEQUEUE (S)
       r ← r[q]
       r[q] ← 0̄
       for each e ∈ E[q]
          do if d[n[e]] ≠ d[n[e]] ⊕ (r ⊗ w[e])
             then d[n[e]] ← d[n[e]] ⊕ (r ⊗ w[e])
                  r[n[e]] ← r[n[e]] ⊕ (r ⊗ w[e])
                  if n[e] ∉ S
                     then ENQUEUE (S, n[e])
d[s] ← 1̄
```

35. The method of claim 30, wherein the step of computing the ε-closure for each state "p" further comprises computing each the ε-closure according to the following equation:

$$C[p] = \{(q,w) : q \in \epsilon[p], d[p,q] = w \in K - \{\bar{0}\}\}.$$

36. The method of claim 35, wherein the step of modifying outgoing transitions of each state "p" further comprises modifying the outgoing transitions of each state p according to the following procedure:

```
for each p ∈ Q
  do E[p] ← {e ∈ E[p] : i[e] ≠ ε}
    for each (q,w) ∈ C[p]
        do E[p] ← E[p] ∪ {(p,a,w ⊗ w',r) : (q, a, w', r) ∈ E[q], a ≠ ε}
          if q ∈ F
             then if p ∉ F
                then F ← F ∪ {p}
                    ρ[p] ← ρ[p] ⊕ (ω ⊕ ρ[q])
```

37. The method of claim 36, wherein a state is a final state if some state "q" within a set of states reachable from "p" via a path labeled with an empty string is final and the final weight is then: $\rho[p] = \bigoplus_{q \in \epsilon[p] \cap F} (d[p,q] \otimes \rho[q])$.

38. The method of claim 37, further comprising:

performing a depth-first search of the transducer A after removing the empty strings.